Building need-based systems for complex hostile situations

Mikael Gullberg, M Gullberg Systemkonsult
m_gullberg@acm.org

Olle Olsson, SICS
olleo@sics.se

0. Abstract

The concept of need-based systems is that they are not instantiated until they are brought to use. Such systems are either designed, assembled and instantiated for very concrete needs, alternatively built on speculation. They are used for evaluation, training or for real. In most cases they are not instantiated, though possibly brought into use as needed. How should needs be expressed? Needs are expressed in terms of results and effects. They may be simple or complex, where complex needs can be seen as structured compositions of more basic needs. When facing a hostile situation, it is critical to have predicted all needs necessary for resolving the given assignment. The complexity of needs, and of their interdependencies may appear to be overwhelming, and it is important to find optimal solutions to the problem at hand. What is needed is a knowledge-based conceptual model that describes the inter-relation of needs, predictions and effects. This paper proposes a number of viewpoints and tools to be applied to the construction of need-based systems—viewing needs as a resource economy, as a non-linear dynamical system, making use of game theory, decision theory, and risk management. Models for needs and predictions, and the ways they are utilized, extends current systems engineering methods.

1. Introduction

By a system, we mean an entity that can be described as a set of components (resources) combined into a structure. A system enables actions to be taken, actions that contribute to a stated objective. For a *need-based system*, the motivation for the system resides in stated needs, needs that are deemed to be critical for the context in which the system is to exist. The space of needs can be seen as a spectrum, from very general and long-lived needs, to very specific and short-lived needs.

Traditional views on systems have emphasized the long-lived and general needs. That is, systems are defined as stable entities that should be able to cater for a very broad range of concrete needs. A more recent trend is to approach system building as something that should be targeted to specific circumstances, i.e., should be addressing specific needs. By adopting this approach, one can achieve a better use of resources, as well as a higher expectation that the system will be able to match the needs it will encounter during its life-time.

The concept of need-based systems introduces certain challenges of a methodological character. In what ways do need-based systems differ from other kinds of systems, and how can a process for building need-based systems be structured? What different types of need-based systems can one identify, and what kind of impact will such distinctions have on the methods for system building? What criteria can be used for evaluating a need-based system, and how can such criteria guide the process of constructing such systems?

In this paper we describe a framework for understanding the concept of need-based systems, identify some critical characteristics of such systems and the contexts in which they exist, and offer a high-level view on the process of building need-based systems. What is here described should be regarded as steps towards a methodology for construction of need-based systems, and we illustrate the approach by highlighting some factors of importance for such a methodology.

2. Main Concepts

The kinds of entities that we call systems always exist in some context, and a system will critically depend on essential characteristics of that context. A system is a socio-technical entity, that contains persons and resources, that has static properties, and that exhibits dynamic behaviors. The three main corner stones on which a system design resides are the *needs* that the system should address; the *rationale* that describes what the objective of the system is, and how the needs are addressed by the system; and the *realization* that describes how the system is created out of other components. Needs are about results and effects – what a system is expected to achieve and what it actually does achieve, expressed in terms of the situation that motivated the design of the system. The needs have an existence that is to a certain degree independent of the system itself – they are regarded as *given* for the system. The realization of the system concerns both the components that it is built out of, and the structure that describes how these components can be composed to create an instantiated system.

We view a system (of the kind we are targeting in this paper) as residing within an enclosing system – that is, systems within an administrative domain are interrelated in a hierarchical fashion. We use the term "super system" of a system S to denote the system that directly contains S (which means that S is a "subsystem" of its super system). The super system itself has associated needs and rationale, and its realization that interrelates its direct subsystems.

There are interdependencies between the super system and the subsystem. The two major ones are that (1) the needs of the subsystem are constrained by the super system, and (2) the rationale of the subsystem depends on the rationale of the super system, mediated by the role the subsystem plays in the design of the super system.

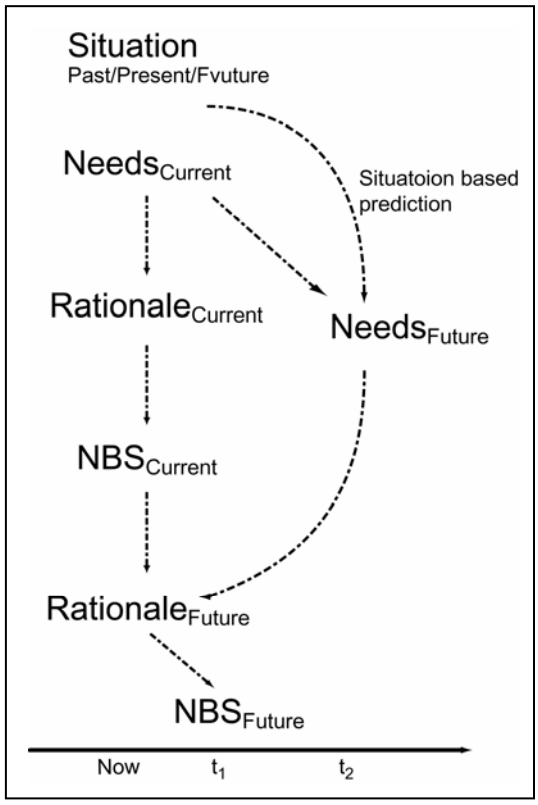

Figure 1. Speculative System Design.

There is a distinction between systems built to address the needs of a current situation and those that address the needs of a possible future situation. In the first case – *reactive* systems building – the situation is present, and the associated needs can be stated. In the second case – *predictive* systems building – we are reasoning about a possible future situation – we are currently not in that situation, but we may at some point in the future be in that situation. These two cases exhibit different characteristics. The reactive case is characterized by a situation that is present, and hence can be inspected to determine what its properties are. Furthermore, the available resources (for system realization) are known, and the time available for designing and instantiating the system may be tightly constrained.

In contrast, the predictive case is characterized by a situation that cannot be described in complete details, and we may not be quite sure about what resources we will have at our disposal, when (and if) the situation occurs and the corresponding system must be instantiated. On the other hand, it is often the case that the time available for designing

for this situation is not so tightly constrained, and more effort can be spent on identifying an optimal system.

This is illustrated in figure 1, where the present situation, its history, and possible succeeding future situations, are taken into account when predicting what likely situations will follow, and what needs they imply. Current needs and envisioned future needs lead to a rationale for a future system, which will then serve as a basis for designing a system adequate for expected future needs.

Two additional properties of systems are critical for understanding what need-based systems are. The first concept is *capacity*, which covers the scope of what the system is *able* to do. A system, in the sense of this paper, is an entity that is able to act, able to have effects in the world. It is constructed out if components (passive resources or active mechanisms) and when the system is used, it will have certain effects. The capacity is whatever can be achieved by the system, whether initially intended or not.

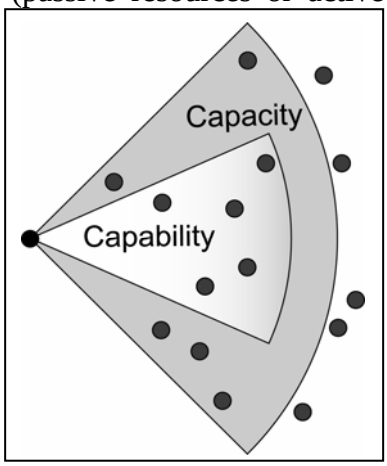

Figure 2. Capacity and capability & situation space

The second concept is *capability*, which is the scope that just includes what the system is *intended* to do. By definition, the capability is a subset of the capacity. As figure 2 illustrates, capability is designed to be adequate for the target situation, but that situation could evolve into other situations, and it is to be expected that the designed capability is adequate for many of them. The capacity of the system can be larger than the capability, and hence the system could handle a wider range of situations, though not all that possibly could occur. Capacity, as well as capability, has its limits, and therefore there are situations that fall beyond the scope of what the system can handle.

Finally, we introduce the concept of *hostile situation*. A situation is regarded as hostile for a system if it presents significant threats to the viability of the system as a provider of its capability. This means that when the system is in such a situation, there is a risk that forces not within the control of the system may exert such a strong influence on the system that it does not deliver as intended. An example of a hostile situation is when an opponent can take actions with the intension of incapacitating our system. Another example is where climatic properties may be so harsh that the redundancy margins of the system disappears and may prevent the system for working as expected.

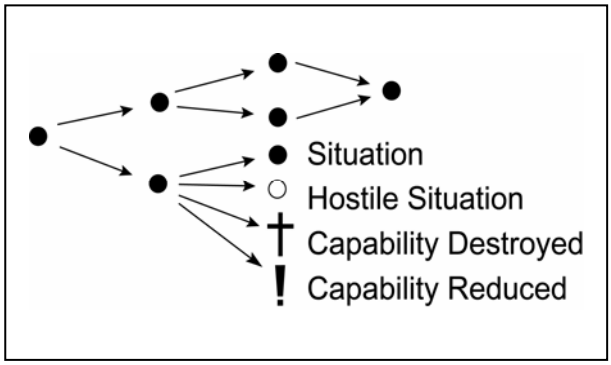

Figure 3. Situation types and transitions.

Figure 3 describes possible situation transitions for a system. The arrows denote how situations may be succeeded by other situations. The actual sequence of situations encountered by a system corresponds to one path through the graph. Some paths lead to success, meaning the system encounters manageable situations during its lifetime. Other paths lead to hostile situations, and this may cause the system to fail – meaning its capability has been reduced or even destroyed.

Figure 4 provides another view, describing how hostile situations may have an impact on the system. In one case, hostile situations below line A have a negative impact ion the

capability (and hence the capacity) of the system. In the second case, hostile situations below line B have a negative impact on the capacity of the system, but not on its capability.

An important property of some of the concepts above is complexity, which is a qualification that can be of critical importance. Informally speaking, high complexity typically leads to complications in the tasks that involve such entities, while low complexity may simplify analysis and design tasks. Two main types of entities that may be complex are systems and situations. Systems complexity arises mainly out of the dependencies between the components of the system, possibly combined with the variation between the types of components. Complexity in systems causes difficulties in determining in detail what a system may achieve, which has an impact on design decisions during the design process (complication in the synthesis task). Furthermore, system complexity may lead to brittleness in the system itself, which may can cause drastic failures in capability provision. Complexity in situations causes problems in determining detailed properties of the situation itself, as well as determining how situations may evolve (complications in the analysis task).

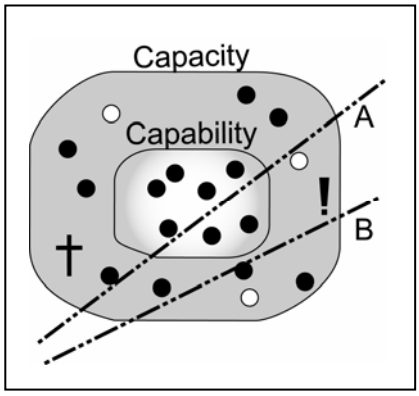

Figure 4. Situation impact.

In a nut-shell, the challenge is; how can needs, rationale, and available or possible components be used to define a system that has a capability that matches the needs and fulfills the rationale, and that has a capacity that sufficiently underpins the capability in situations that may deviate from what was expected.

3. Process tasks

In this section, we discuss critical tasks in the process of designing a system. The main generic tasks are *orchestration*, *navigation*, *optimization*, and *prediction*.

Orchestration

Orchestration is about how a system is structured as a combination of a set of components (building blocks). The challenge of orchestration is that it generates combinatorial complexity in the design process, when one tries to identify candidate system designs for stated needs. Resources (of all types) are the building blocks upon which a system design is based. Such resources can be pre-existing, with potential static properties and dynamic behaviors. Resources can be characterized by a resource ontology, describing essential properties of various resource categories. For instance, certain resources correspond to mass nouns, and they can be described by quantities (e.g., fuel). Resources may be consumable or they may be re-usable. For instance, fuel is consumable, while a port can be re-usable. Certain compositions of resources in a structure may be permitted (possible, meaningful) while other may be forbidden (impossible, meaningless). For instance, fuel may be combined with containers, as it makes sense to store fuel in containers, while combining fuel with weather is meaningless. This turns the problem of combining resources into a constraint problem – any combination must satisfy the constraints specified for the resources involved.

Another type of resource is system – systems can be used as subsystem components when creating a super system. Subsystem designs may or may not pre-exist. A pre-existing system design is a building block with known properties and behaviors. For such systems, there are constraints that define what conditions must be fulfilled if this system is used as a building block in a larger system.

The concept of "not pre-existing system" covers the circumstances when one can design a new system to serve as a building block in the super system. This approach can offer more degrees of freedom, compared to use of pre-existing systems, as we are not constrained by the characteristics exhibited by the fixed set of existing system. When designing a system at a given level one can specify a subsystem that exhibits those properties and behaviors that provide the support needed for the system being built at the given level. The specification of such a "hypothetical" (or ideal) systems creates a system design problem at a lower level – the subsystem level – and that system design problem is solved at that lower level. Each system design task requires knowledge about what resources may be utilized as building blocks for the design task – those resources are provided by the super system.

Figure 5 depicts a simple example of a three-level hierarchy of systems. Each box corresponds to a system, with its rationale, its needs, and the corresponding need-based system. A rationale provided at a higher level influences the decision on what the rationale of a lower level expresses. Moreover, a system is associated with a life cycle (indicated by the "progress bar" in each box in the figure), covering all managed states of life of the system – from the idea of the system to its dismantling and archiving.

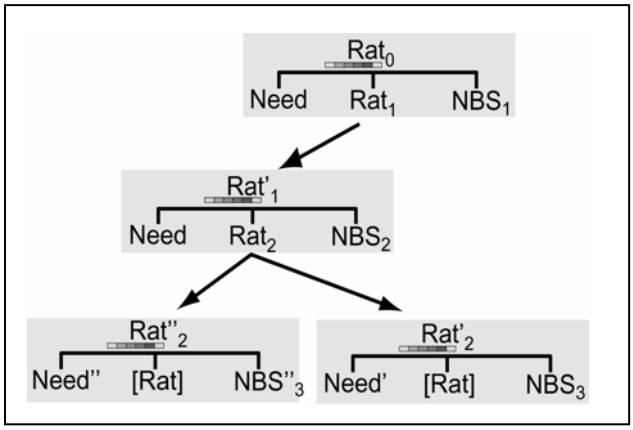

Figure 5. Hierarchy of systems rationale disposition

Orchestration is essentially a complex resource-management problem. Decision making on such problems is non-trivial, as one has to ascertain that system proposals are inherently consistent, that there are enough resources to instantiate the system, and that the system should address the given needs.

Verification is an important issue for orchestration. For system designs to be acceptable, they must satisfy certain explicitly stated criteria. These criteria correspond to required properties of system designs, properties that target other properties than the capability of the system.

Navigation

The set of all system designs create a design space – a space consisting of all possible system designs. There are three main approaches to goal-directed design: (1) bottom-up design, (2) top-down design, and (3) transformation of system designs. In the first approach – bottom-up design – one creates a system design by incrementally adding components to an incomplete design, ultimately ending up with a complete design. In the second approach – top-down design – one starts with an abstract design that covers the needs, and then incrementally replacing lower level abstractions with more specific constructs, ultimately ending up with a complete design. The third approach – transformation design – takes some candidate design as starting point and by iteratively transforming the design, getting a sequence of complete designs that approaches a design that meets the desired requirements.

Another approach is one that is based on pre-defined skeletal re-usable designs (or design templates). Here the challenge is finding suitable re-usable designs that can be adapted to serve as components in the evolving system design

These approaches, and other alternative approaches, have subtasks that can be seen as navigation tasks – navigating in the space of components, reusable design templates, and designs themselves. Well-designed navigation tools can provide invaluable support for the decision-making that takes place during the design process.

Optimization

For a given rationale and given needs, there may be several system designs that are candidate designs – that is they satisfy the rationale and meet the needs. From one point of view, they are all acceptable, but they may differ in various ways. Some of these differences may be insignificant, but other differences may be regarded as important for the use of the system.

Selection among candidate designs is an optimization problem – optimizing with respect to a set of optimization criteria. In a more general sense, any selection of a design can be seen as optimization. Properties of designs form a multidimensional space, and the goal function maps points in this space to the properties of the rationale.

Examples of optimization can be the expected amount of fuel consumed during operations, or the robustness w.r.t. changes in the situation.

Whether to optimize or not, and to what extent optimization should be pursued, that depends critically on the circumstances of the design task. When one has to design a system to meet the needs of the current situation, time may be a scarce resource, so it might be better not to pursue full optimization – it is better to have a reasonable system instantiated now, than to have an optimal system later. If the circumstances are such that we are designing systems for possible future instantiations, then it may be suitable to perform extensive optimizations.

Prediction

Situations are dynamic, and may evolve into other more or less dissimilar situations. The concrete situation the system will find itself in sometime in the future may differ significantly from the targeted situation in critical respects. It is therefore possible that the system will not adequately address the needs of the situation it actually will be situated in, even though it did match the needs of the situation that was initially envisioned.

The obvious way to decrease the risk of designing a system that does not match a future situation is to predict possible future situation that the present situation may evolve into.

How situations are likely to evolve depends on characteristics of the situation itself, and some prototypical cases will be mentioned here. Firstly, the total situation (the "universal situation") in itself may be in a stable state, and this gives a high probability that the situation will not drastically change. Secondly, as part of the total system we may have *nature* as a critical component, and this may, in a non-discriminatory way, cause the total system to evolve in certain ways. For instance, it weather is critical, one can predict with reasonable accuracy what the weather will be in the short-term (up to, say, five days), but in the long-term, the weather is a chaotic system (Lorenz, 1993), and cannot be predicted in practice. Thirdly, we may have an adversary as part of the total system, an adversary that will try to force the total system to evolve in ways that is to his advantage and to our disadvantage.

As the above illustrates, there are different kinds of forces influencing how situations evolve, and they may have to be handled in different ways if reliable predictions are to be made.

Predictions about how situations may evolve, gives indications about the characteristics of future situations. Based on knowledge about possible future situations, one can specify a suitable rationale that captures essential knowledge about the situation and what we want a system to achieve in the situation.

The rationale expresses a value function that tells us what values should be achieved. It is a foundation for the specification of the needs of the system. The needs will then influence what capability we would require for these needs. Moreover, a system offering this capability will possess a certain capacity.

The step from rationale to needs involves risk management. This covers issues like risk estimation and risk valuation.

4. Methods and tools

A conceptual framework for system design was described above, in which critical design challenges were identified. In this section, we illustrate how various approaches can be used as methods and tools in the process of developing need-based systems.

Predictions concern determining future causes and effects. The most general entity to make predictions about is the situation. There are many methods for predicting future states of various systems, well-known in engineering and physics. We here just want to emphasize another approach that can be applied to situations where an adversary is present – theory and methods for non-cooperative games. The main objective is to determine possible action strategies that will optimize the outcome. For instance, the minimax method gives the strategy to follow in order to control the damage one can be subjected to; see (Luce and Raiffa, 1989) (Shubik, 1985) (Blackwell and Girshick, 1979). Another approach concerns co-operative games, where different parties participating in a shared situation can create a win-win outcome; see (Axelrod, 1984) (Axelrod, 1997).

Decision theory covers a wide range of methods and tools. Classical approaches; see (Fishburn, 1972), have in recent years been complemented by Bayesian methods; see (Smith, 1988).

Reasoning about values is related to theories of utility and preferences. There a growing body of theories and methods is emerging; see (Fishburn, 1988).

Economy-inspired approaches have been applied to situations where classical monetary systems play no role, but where there are other ways of exchanging resources that have value. For instance, the concept of an artificial market where “trading” is being done and artificial values are exchanged have been applied to resource distribution problems. Likewise, market-like mechanisms have been applied to decentralised optimization problems, where a common optimal solution emerges from self-interested actors that trade artificial resources; see (Clearwater, 1996).

Situations that have strong fluent properties can be seen as dynamical systems. Dynamical systems theory gives a foundation for modeling highly fluent systems, in order to determine their evolution over time. Knowledge about the dynamics of a system can provide the basis for understanding how to control that system. Concepts from dynamical systems theory that are relevant for the purpose at hand are stability, attractor, chaos, and cycles; see (Strogatz, 2001).

The situations that embedding the system in focus are also in a flux, sometimes in a antagonistic or even unmercifully changes. The methods and tools suggested for managing the systems are a set of processes e.g. risk management, resource management, validation and so forth. This set is described ISO 15288 *System Life Cycle Processes* and arranged in an architecture framework like FMA, Swedish defence Architecture.

5. Summary and Conclusions

This paper presents a view of the construction of need-based systems. This view is based on a framework, that characterizes need-based systems in terms of a set of critical properties, corresponding to important tasks in the process of developing such systems. We have also indicated what kinds of methods and tools may be used within such tasks.

It is increasingly important to create systems that target specific needs in a world where situations are highly dynamic. This implies that systems must be flexible enough to cater for the cases where the current situation evolves into situations that may be significantly different from what was expected or desired. Furthermore, systems should be created using no more resources than necessary to match the needs that are at hand and needs that may arise.

Need-based systems that have a capability to evolve are one way of meeting such requirements. For this to be a feasible approach, it is necessary to understand the relationship between systems, situations, and needs, as well as how evolution in situations can be matched by systems that are able to adapt to such changes. The framework presented in this paper provides a preliminary model of how this can be achieved.

6. References

R Axelrod (1984). *The Evolution of Cooperation*. New York: Basic Books, 1984.

R. Axelrod (1997). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton, NJ: Princeton University Press, 1997.

D. A. Blackwell and M. A. Girshick (1979). *Theory of Games and Statistical Decisions*. Dover Publications, 1979.

S.H. Clearwater (1996). *Market-Based Control - A Paradigm for Distributed Resource Allocation*, World Scientific Publishing, 1996.

P.C. Fishburn (1972). *Mathematics of Decision Theory*. Mouton, 1972.

P.C. Fishburn (1988). *Nonlinear Preference and Utility Theory*, Johns Hopkins, 1988.

Lorenz, E.N. (1993). *The Essence of Chaos*. University of Washington Press, Seattle, WA, US, 1993.

Luce, R.D. and H. Raiffa (1989). *Games and Decisions: Introduction and Critical Survey*. Dover Publications; Reprint edition, 1989.

M. Shubik (1985): *Game Theory in the Social Sciences, Vol. 1: Concepts and Solutions*. The MIT Press; Reprint edition, 1985.

J.Q. Smith (1988). *Decision Analysis: A Bayesian Approach*. Chapman and Hall. 1988.

S. H. Strogatz (2001): *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Perseus Books Group; 1st edition, 2001.

7. Biographies

Mr. Olle Olsson, works at Swedish Institute of Computer Science (SICS), a national research institute in Information and Communication Technologies (ICT). His research interests are conceptual modeling, semantics, formal methods, and architectural issues.

Mr. Michael Gullberg, works as an independent consultant. Have been working with different systems engineering topics in the defense are for more than 15 years. Currently working with FMV, with enterprise architecture for defence systems. His main research interests are knowledge representation, semantics, formal conceptual models and system architectural questions